TAREA PARA DWEC04 https://github.com/JuanjoMayorga/Mayorga_Usero_JuanJose_DWEC04.git

UT04 Práctica 2: Galería de imágenes

Vamos a construir la estructura de objetos necesaria para implementar una galería de imágenes en una aplicación JavaScript.

A continuación, se detalla los objetos que debemos implementar junto con su funcionalidad. Ten en cuenta que deberás implementar también la estructura de objetos necesaria para gestionar las excepciones que genere la aplicación.

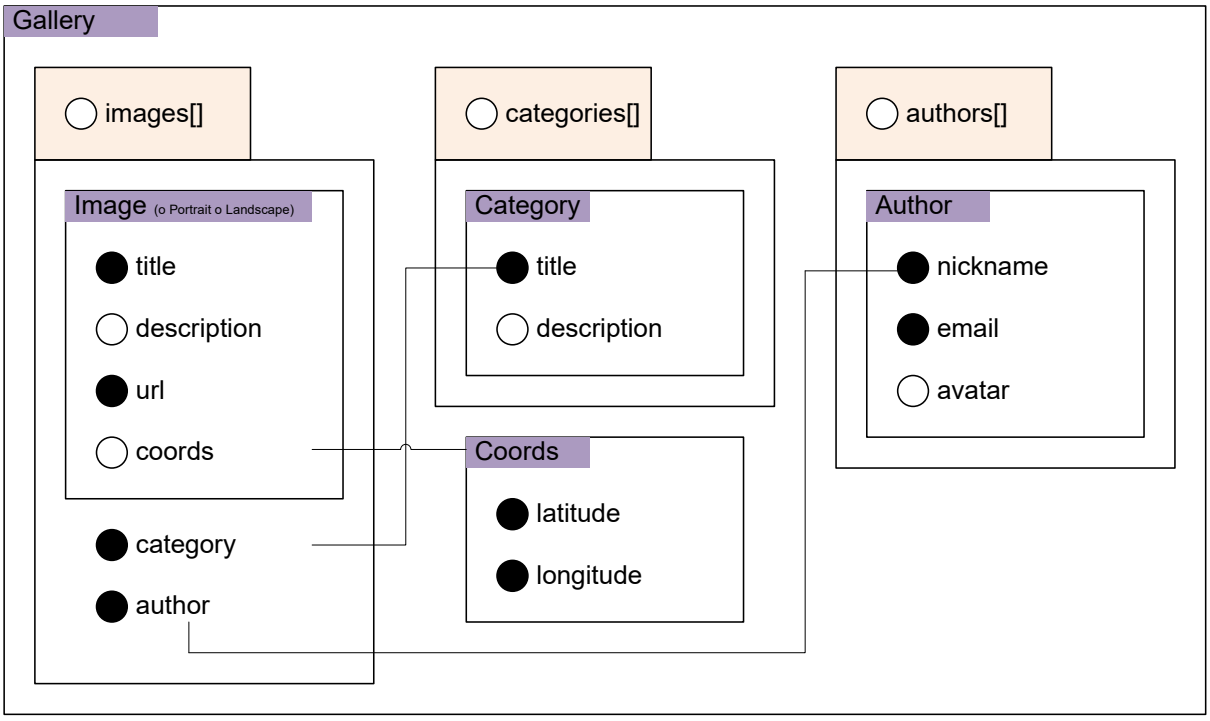
Para este ejercicio he pensado que la mejor manera de organizar la información es pensar en un objeto Gallery, que sería como una base de datos. Este objeto tiene tres arrays: uno de imágenes, otro de categorías y otro de autores. Cada uno de estos arrays sería análogo a una tabla en una base de datos.

De esta manera, el array categorías tendría un título, de carácter obligatorio, que sería una especie de clave primeria.

En el array autores habría un nickname ejerciendo de clave primaria, y otros atributos,

El array de imágenes es el más complejo, porque contiene referencias a los otros dos. En el enunciado del problema se dan un par de opciones, pero he pensado que la mejor forma de tratar este array es "importando", a modo de clave foránea, las claves primarias de los otros dos arrays (título de la categoría y nickname del autor) y asociárselas al objeto imagen que queremos almacenar.

Gráficamente, la estructura de la información sería así:



1. Objeto Gallery

Este objeto mantendrá el estado de la galería. La información que debe mantener es:

- **Título de la galería**
- **Las imágenes que hayamos añadido a la galería. Implementado mediante un array de objetos imágenes.**
- **Las categorías de imágenes que tengas en la galería. Implementado mediante un array de objetos categoría.**
- **Los autores de las diferentes imágenes. Implementado mediante un array.**
- **Categoría por defecto. En caso de no utilizar ninguna categoría a una imagen se utilizará esta categoría.**
- **Autor por defecto. En caso de no utilizar ningún autor al añadir una imagen, se utilizará este autor.**

Los métodos que debéis implementar para este objeto son:

Método	Funcionalidad	Argumentos	Retorno	Excepciones
Getter/setter title	Para la propiedad Title	String	String	- El título no puede ser vacío
Getter Categories	Devuelve el array de objetos categorías	-	[] Category	-
Getter Authors	Devuelve un array con los autores	-	[] Author	-
addCategory	Añade una nueva categoría	Objeto Category	Number con el nº de elementos	- La categoría no puede ser null. - La categoría ya existe
removeCategory	Elimina una categoría	Objeto Category	Number con el nº de elementos	- La categoría no está registrada
addImage	Añade una nueva imagen a una categoría con un autor.	- Objeto Image - Objeto Category - Objeto Author	Number con el nº de elementos	- La imagen no puede ser null.
removeImage	Elimina una imagen	Objeto Image	Number con el nº de elementos	- La imagen no existe.
getCategoryImages	Devuelve todas las imágenes de una determinada categoría	Objeto Category	[]Images	- La categoría no puede ser null.
addAuthor	Añade un nuevo autor a la Galería	Objeto Author	Number con el nº de autores.	- El autor no puede ser nulo
removeAuthor	Eliminar un autor	Objeto Author	Number con el nº de elementos	- Author no existente.
getAuthorImages	Devuelve todas las imágenes añadidas por un autor.	Objeto Author	[]Images	- El autor no puede ser null.
getPortraits	Devuelve todas las imágenes de tipo retrato	-	[]Portrait	-
getLandscapes	Devuelve todas las imágenes de tipo apaisado	-	[]Landscape	-

Gallery.js

```
/*
Juan José Mayorga Usero
CFGS Desarrollo de Aplicaciones Web
I.E.S. Maestre de Calatrava - Ciudad Real
Desarrollo Web en Entorno Cliente
UT04 Práctica 2
*/

function Gallery(title)
{
    if (!(this instanceof Gallery))
    {
        throw new ConstructorInvalido();
    }

    // title es un atributo obligatorio.
    if(!title)
    {
        throw new TituloVacio();
    }

    // Defino title y los arrays de imágenes, categorías y autores.
    let _title = title;
    let _images = [];
    let _categories = [];
    let _authors = [];

    // Getters y Setters de los atributos.
    Object.defineProperty(this, 'title',
    {
        get:function()
        {
            return _title;
        },
        set:function(value)
        {
            if (!value)
            {
                throw new TituloVacio();
            }
            _title = value;
        }
    });

    Object.defineProperty(this, 'categories',
    {
        get:function()
        {
            return _categories;
        },
    });

    Object.defineProperty(this, 'authors',
    {
        get:function()
        {
            return _authors;
        },
    });

    Object.defineProperty(this, 'images',
    {
        get:function()
```

```
        {
            return _images;
        },
    });
}

Gallery.prototype.constructor = Gallery;

// Añade una nueva categoría.
Gallery.prototype.addCategory = function(category)
{
    try
    {
        if (!(category instanceof Category))
        {
            throw new CategoriaInvalida();
        }
        // Excepción: La categoría no puede ser null
        if (category==null)
        {
            throw new CategoriaNull();
        }
        // Excepción: La categoría ya existe.
        this.categories.forEach(function(entry)
        {
            if(entry.title==category.title)
            {
                throw new CategoriaExiste();
            }
        });
        // Este código era válido planteando el ejercicio según la solución. Cambié de idea
.
        /*category = {
            category : category,
            images: []
        }*/
        this.categories.push(category); //Utilizamos los métodos de array para gestionar la
 lista.
        return this.categories.length; // Devolvemos el tamaño.
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

// Elimina una categoría.
Gallery.prototype.removeCategory = function(category)
{
    try
    {
        // Primero hay que buscar la categoría que queremos quitar.
        // Para ello buscamos su índice comparando el título de la categoría buscada con el
 atributo categoría de las imágenes del array.
        // Según lo descrito en la estructura de datos.
        var index = -1;
        for (var i = 0; i < this.categories.length; i++)
        {
            /*if (this.categories[i].category.title == category.title)*/
            if (this.categories[i].title == category.title)
            {
                index = i;
                break;
            }
        }
```

```
        // Excepción: La categoría no está registrada.
        if(index==-1)
        {
            throw new CategoriaNoRegistrada();
        }
        this.categories.splice(index, 1);
        return this.categories.length;
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

// Añade una nueva imagen a una categoría con un autor.
Gallery.prototype.addImage = function(image, category, author)
{
    try
    {
        if (!(image instanceof Image))
        {
            throw new ImagenInvalida();
        }
        if (!(category instanceof Category))
        {
            throw new CategoriaInvalida();
        }
        if (!(author instanceof Author))
        {
            throw new AutorInvalido();
        }
        // Excepción: La imagen no puede ser null
        if (image==null)
        {
            throw new ImagenNull();
        }
        // Busco si la imagen a introducir ya está en el array.
        // En tal caso, lanzo una excepción.
        var indexImagen = -1;
        for (var i = 0; i < this.images.length; i++)
        {
            if (this.images[i].image.title == image.title)
            {
                indexImagen = i;
                break;
            }
        }
        // Excepción: La imagen ya existe.
        if(indexImagen!=-1)
        {
            throw new ImagenExiste();
        }
        // Compruebo si la categoría existe en el array de categorías.
        var indexCategoria = -1;
        for (var i = 0; i < this.categories.length; i++)
        {
            if (this.categories[i].title == category.title)
            {
                indexCategoria = i;
                break;
            }
        }
        if(indexCategoria == -1)
        {
            throw new CategoriaNoRegistrada();
```

```
        }
        // Compruebo si el autor existe en el array de autores.
        var indexAutor = -1;
        for (var i = 0; i < this.authors.length; i++)
        {
            if (this.authors[i].nickname == author.nickname)
            {
                indexAutor = i;
                break;
            }
        }
        if(indexAutor == -1)
        {
            throw new AutorNoExiste();
        }
        /* Si no hay problemas con los datos a introducir, creo un objeto compuesto por el
objeto Image y las propiedades
        nickname de author y title de category.
        Esto se almacena en el array de imágenes.
        */
        image = {
            image : image,
            author: this.authors[indexAutor].nickname,
            category: this.categories[indexCategoria].title
        }
        /*this.categories[indexCategoria].images.push(image);*/
        this.images.push(image);
        return this.images.length; // Devolvemos el tamaño.
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

// Elimina una imagen.
Gallery.prototype.removeImage = function(image)
{
    try
    {
        // Compruebo si la categoría a eliminar existe.
        var index = -1;
        for (var i = 0; i < this.images.length; i++)
        {
            if (this.images[i].image.title == image.title)
            {
                index = i;
                break;
            }
        }
        // Excepción: La imagen no existe.
        if(index==-1)
        {
            throw new ImagenNoExiste();
        }
        this.images.splice(index, 1);
        return this.images.length;
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

// Devuelve todas las imágenes de una determinada categoría.
```

```
Gallery.prototype.getCategoryImages = function(category)
{
    try
    {
        // Creo una variable para almacenar los datos de retorno.
        let _imagesCategory = [];
        // Recorro el array de imágenes buscando las imágenes de esa categoría.
        // La estructura de datos que he ideado facilita mucho la labor.
        var elementosEncontrados = 0;
        for (var i = 0; i < this.images.length; i++)
        {
            if (this.images[i].category == category.title)
            {
                _imagesCategory.push(this.images[i]);
                elementosEncontrados++;
            }
        }
        // Excepción: La categoría no está registrada.
        if(elementosEncontrados==0)
        {
            throw new CategoriaNoRegistrada();
        }
        return _imagesCategory;
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

// Añade un nuevo autor a la galería.
Gallery.prototype.addAuthor = function(author)
{
    try
    {
        if (!(author instanceof Author))
        {
            throw new AutorInvalido();
        }
        // Excepción: El autor no puede ser null
        if (author==null)
        {
            throw new AutorNull();
        }
        // Excepción: El autor ya existe.
        this.authors.forEach(function(entry)
        {
            if(entry.nickname==author.nickname)
            {
                throw new AutorExiste();
            }
        });
        this.authors.push(author); //Utilizamos los métodos de array para gestionar la list
a.
        return this.authors.length; // Devolvemos el tamaño.
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

// Eliminar un autor.
Gallery.prototype.removeAuthor = function(author)
{
```

```
    try
    {
        var index = -1;
        // Compruebo si existe el autor que queremos eliminar.
        for (var i = 0; i < this.authors.length; i++)
        {
            if (this.authors[i].nickname == author.nickname)
            {
                index = i;
                break;
            }
        }
        // Excepción: El autor no existe.
        if(index==-1)
        {
            throw new AutorNoExiste();
        }
        this.authors.splice(index, 1);
        return this.authors.length;
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

// Devuelve todas las imágenes añadidas por un autor.
Gallery.prototype.getAuthorImages = function(author)
{
    try
    {
        // Creo una variable para almacenar los datos de retorno.
        // Recorro el array de imágenes buscando las imágenes de ese autor.
        // La estructura de datos que he ideado facilita mucho la labor.
        let _imagesAuthor = [];
        var elementosEncontrados = 0;
        for (var i = 0; i < this.images.length; i++)
        {
            if (this.images[i].author == author.nickname)
            {
                _imagesAuthor.push(this.images[i]);
                elementosEncontrados++;
            }
        }
        // Excepción: El autor no existe.
        if(elementosEncontrados==0)
        {
            throw new AutorNoExiste();
        }
        return _imagesAuthor;
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

// Devuelve todas las imágenes de tipo retrato.
Gallery.prototype.getPortraits = function()
{
    try
    {
        // Creo una variable para almacenar los datos de retorno.
        let _portraits = [];
        // Recorro el array de imágenes en busca de los elementos Portrait.
```

```
        for (var i = 0; i < this.images.length; i++)
        {
            if (this.images[i].image instanceof Portrait)
            {
                _portraits.push(this.images[i]);
            }
        }
        return _portraits;
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

// Devuelve todas las imágenes de tipo apaisado.
Gallery.prototype.getLandscapes = function()
{
    try
    {
        // Creo una variable para almacenar los datos de retorno.
        let _landscapes = [];
        //  Recorro el array de imágenes en busca de los elementos Landscape.
        for (var i = 0; i < this.images.length; i++)
        {
            if (this.images[i].image instanceof Landscape)
            {
                _landscapes.push(this.images[i]);
            }
        }
        return _landscapes;
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

/* He añadido una función imprimir para mostrar el contenido de las galerías.
Esto se debe a que si las muestro con el comando console.log, saldrá el estado final del ob
jeto Galleru, no su estado en el momento que se invoca.
 */
Gallery.prototype.imprimir = function()
{
    let texto = "images : \n";
    this.images.forEach(function(entry)
    {
        texto = texto + "{\n\timage : \n\t{ \n\t\ttitle : "+entry.image.title+"\n\t\tdescri
ption : "+entry.image.description+"\n\t\turl : "+entry.image.url+"\n\t\tcoords : \n\t\t{\n\
t\t\tlatitude : "+entry.image.coords.latitude+"\n\t\t\tlongitude : "+entry.image.coords.lon
gitude+"\n\t\t}\n\t}\n\tauthor : "+entry.author+"\n\tcategory : "+entry.category+"\n}\n";
    });
    texto = texto + "categories : \n";
    this.categories.forEach(function(entry)
    {
        texto = texto + "{\n\ttitle : "+entry.title+"\n\tdescription : "+entry.description+
"\n}\n";
    });
    texto = texto + "authors : \n";
    this.authors.forEach(function(entry)
    {
        texto = texto + "{\n\tnickname : "+entry.nickname+"\n\temail : "+entry.email+"\n\ta
vatar : "+entry.avatar+"\n}\n";
    });
    console.log(texto);
```

```
}
```

2. Objeto Category

Con este objeto podemos crear la estructura de categorías de imágenes de la galería. Sus propiedades serán:

- **Title: Con el título de la categoría.**
- **Description: Con la descripción de la misma.**

Sus métodos serán los habituales getter y setter de estas propiedades, aunque el título no podrá ser vacío.

Category.js

```
/*
Juan José Mayorga Usero
CFGS Desarrollo de Aplicaciones Web
I.E.S. Maestre de Calatrava - Ciudad Real
Desarrollo Web en Entorno Cliente
UT04 Práctica 2
*/

function Category(title, description = "")
{
    if (!(this instanceof Category))
    {
        throw new ConstructorInvalido();
    }

    if(!title)
    {
        throw new TituloVacio();
    }

    let _title = title;
    let _description = description;

    Object.defineProperty(this, 'title',
    {
        get:function()
        {
            return _title;
        },
        set:function(value)
        {
            if (!value)
            {
                throw new TituloVacio();
            }
            _title = value;
        }
    });

    Object.defineProperty(this, 'description',
    {
        get:function()
        {
            return _description;
        },
        set:function(value)
```

```
        {
            _description = value;
        }
    });
}

Category.prototype.constructor = Category;
```

3. Objeto Image

Objeto que permitirá añadir imágenes a la galería. Tendrá las siguientes propiedades:

- **Title: Título de la imagen. Obligatoria.**
- **Description: Descripción de la imagen**
- **URL: Dirección donde está ubicada la imagen. Obligatoria.**
- **Coords: Coordenadas donde se tomó la imagen.**

Como métodos tendrá los getter y setter habituales.

Image.js

```
/*
Juan José Mayorga Usero
CFGS Desarrollo de Aplicaciones Web
I.E.S. Maestre de Calatrava - Ciudad Real
Desarrollo Web en Entorno Cliente
UT04 Práctica 2
*/

function Image(title, description, url, coords = "")
{
    if (!(this instanceof Image))
    {
        throw new ConstructorInvalido();
    }

    if(!title)
    {
        throw new TituloVacio();
    }

    if(!description)
    {
        throw new DescripcionVacia();
    }

    if(!url)
    {
        throw new URLVacia();
    }

    let _title = title;
    let _description = description;
    let _url = url;
    let _coords = coords;

    Object.defineProperty(this, 'title',
    {
        get:function()
        {
            return _title;
```

```
        },
        set:function(value)
        {
            if (!value)
            {
                throw new TituloVacio();
            }
            _title = value;
        }
    });

    Object.defineProperty(this, 'description',
    {
        get:function()
        {
            return _description;
        },
        set:function(value)
        {
            if (!value)
            {
                throw new DescripcionVacia();
            }
            _description = value;
        }
    });

    Object.defineProperty(this, 'url',
    {
        get:function()
        {
            return _url;
        },
        set:function(value)
        {
            if (!value)
            {
                throw new URLVacia();
            }
            _url = value;
        }
    });

    Object.defineProperty(this, 'coords',
    {
        get:function()
        {
            return _coords;
        },
        set:function(value)
        {
            _coords = value;
        }
    });
}

Category.prototype.constructor = Category;
```

3.1. Objeto Landscape

Hereda de Image para especializar una imagen apaisada.

No tiene propiedades específicas.

Landscape.js

```
/*
Juan José Mayorga Usero
CFGS Desarrollo de Aplicaciones Web
I.E.S. Maestre de Calatrava - Ciudad Real
Desarrollo Web en Entorno Cliente
UT04 Práctica 2
*/

function Landscape(title, description, url, coords = "")
{
    if (!(this instanceof Landscape))
    {
        throw new ConstructorInvalido();
    }

    Image.call(this, title, description, url, coords);
}

Landscape.prototype = Object.create(Image.prototype);
Landscape.prototype.constructor = Landscape;
```

3.2. Objeto Portrait

Hereda de Image para especilizar una imagen en forma de retrato.

No tiene propiedades específicas.

Portrait.js

```
/*
Juan José Mayorga Usero
CFGS Desarrollo de Aplicaciones Web
I.E.S. Maestre de Calatrava - Ciudad Real
Desarrollo Web en Entorno Cliente
UT04 Práctica 2
*/

function Portrait(title, description, url, coords = "")
{
    if (!(this instanceof Portrait))
    {
        throw new ConstructorInvalido();
    }

    Image.call(this, title, description, url, coords);
}

Portrait.prototype = Object.create(Image.prototype);
Portrait.prototype.constructor = Portrait;
```

4. Objeto Coords

Coordenadas para adjuntar a una imagen. Sus propiedades son:

- **Latitude: Latitud.**
- **Longitude: Longitud.**

Los métodos serán getter y setter, siendo ambas propiedades obligatorias.

Coords.js

```
/*
Juan José Mayorga Usero
CFGS Desarrollo de Aplicaciones Web
I.E.S. Maestre de Calatrava - Ciudad Real
Desarrollo Web en Entorno Cliente
UT04 Práctica 2
*/

function Coords(latitude, longitude)
{
    if (!(this instanceof Coords))
    {
        throw new ConstructorInvalido();
    }

    if(!latitude)
    {
        throw new LatitudVacia();
    }

    if(!longitude)
    {
        throw new LongitudVacia();
    }

    let _latitude = latitude;
    let _longitude = longitude;

    Object.defineProperty(this, 'latitude',
    {
        get:function()
        {
            return _latitude;
        },
        set:function(value)
        {
            if (!value)
            {
                throw new LatitudVacia();
            }
            _latitude = value;
        }
    });

    Object.defineProperty(this, 'longitude',
    {
        get:function()
        {
            return _longitude;
        },
        set:function(value)
        {
            if (!value)
            {
                throw new LongitudVacia();
            }
            _longitude = value;
        }
    });
}
```

```
Coords.prototype.constructor = Coords;
```

5. Objeto Author

Información del autor de la imagen. Sus propiedades son:

- **Nickname: Nombre del autor. Obligatorio.**
- **Email: Correo electrónico. Obligatorio.**
- **Avatar: imagen asociada al usuario.**

Como métodos tendrá los getter y setter habituales teniendo en cuenta las propiedades que son obligatorias.

Author.js

```
/*
Juan José Mayorga Usero
CFGS Desarrollo de Aplicaciones Web
I.E.S. Maestre de Calatrava - Ciudad Real
Desarrollo Web en Entorno Cliente
UT04 Práctica 2
*/

function Author(nickname, email, avatar = "")
{
    if (!(this instanceof Author))
    {
        throw new ConstructorInvalido();
    }

    if(!nickname)
    {
        throw new NombreVacio();
    }
    if(!email)
    {
        throw new EmailVacio();
    }

    let _nickname = nickname;
    let _email = email;
    let _avatar = avatar;

    Object.defineProperty(this, 'nickname',
    {
        get:function()
        {
            return _nickname;
        },
        set:function(value)
        {
            if (!value)
            {
                throw new TituloVacio();
            }
            _nickname = value;
        }
    });

    Object.defineProperty(this, 'email',
    {
```

```
        get:function()
        {
            return _email;
        },
        set:function(value)
        {
            if (!value)
            {
                throw new EmailVacio();
            }
            _email = value;
        }
    });

    Object.defineProperty(this, 'avatar',
    {
        get:function()
        {
            return _avatar;
        },
        set:function(value)
        {
            _avatar = value;
        }
    });
}

Author.prototype.constructor = Author;
```

Para completar el ejercicio, además de la estructura de datos necesito una serie de archivos adicionales:

Mayorga_Usero_JuanJose_DWEC04_UT04-2.html

El archivo .html con el que desplegaré la página web y bajo la que correrá el código JavaScript.

```
<!DOCTYPE html>
<html>
    <head>
        <title>DWEC04 - Práctica 2 - Juan José Mayorga Usero</title>
        <!--
 Utilizo una hoja de estilos que he creado para mejorar la presentación de la Tarea. -->
        <link  rel="stylesheet" type="text/css"  href="Mayorga_Usero_JuanJose_DWEC04_hojade
estilo.css"/>
    </head>
    <body>
        <!-- Creo un div con mis datos y los del módulo, al que aplico un estilo. -->
        <div class="cabecera">
            <h3>CFGS Desarrollo de Aplicaciones Web</h3>
            <h3>I.E.S. Maestre de Calatrava - Ciudad Real</h3>
            <h3>Desarrollo Web en Entorno Cliente</h3>
            <h3>Juan José Mayorga Usero</h3>
        </div>
        <br/>
        <!-- Creo un div con el enunciado del apartado, al que aplico un estilo. -->
        <div class="enunciado">
            <h1>DWEC04</h1>
            <h2>UT04 Práctica 2: Galería de imágenes</h2>
            <h4>Vamos a construir la estructura de objetos necesaria para implementar una g
alería de imágenes en una aplicación JavaScript.</h4>
            <h4>A continuación, se detalla los objetos que debemos implementar junto con su
 funcionalidad. Ten en cuenta que deberás implementar también la estructura de objetos nece
saria para gestionar las excepciones que genere la aplicación.</h4>
```

```
            <h3>1. Objeto Gallery</h3>
            <h4>Este objeto mantendrá el estado de la galería. La información que debe mant
ener es:</h4>
            <h4>- Título de la galería<br/>
                -
 Las imágenes que hayamos añadido a la galería. Implementado mediante un array de objetos i
mágenes.<br/>
                -
 Las categorías de imágenes que tengas en la galería. Implementado mediante un array de obj
etos categoría.<br/>
                -
 Los autores de las diferentes imágenes. Implementado mediante un array.<br/>
                -
 Categoría por defecto. En caso de no utilizar ninguna categoría a una imagen se utilizará
esta categoría.<br/>
                -
 Autor por defecto. En caso de no utilizar ningún autor al añadir una imagen, se utilizará
este autor.
            </h4>
            <h4>Los métodos que debéis implementar para este objeto son:</h4>
            <table>
                <tr class="filacabeza">
                    <td>Método</td>
                    <td>Funcionalidad</td>
                    <td>Argumentos</td>
                    <td>Retorno</td>
                    <td>Excepciones</td>
                </tr>
                <tr class="fila">
                    <td>Getter/setter title</td>
                    <td>Para la propiedad Title</td>
                    <td>String</td>
                    <td>String</td>
                    <td>- El título no puede ser vacío</td>
                </tr>
                <tr class="fila">
                    <td>Getter Categories</td>
                    <td>Devuelve el array de objetos categorías</td>
                    <td>-</td>
                    <td>[] Category</td>
                    <td>-</td>
                </tr>
                <tr class="fila">
                    <td>Getter Authors</td>
                    <td>Devuelve un array con los autores</td>
                    <td>-</td>
                    <td>[] Author</td>
                    <td>-</td>
                </tr>
                <tr class="fila">
                    <td>addCategory</td>
                    <td>Añade una nueva categoría</td>
                    <td>Objeto Category</td>
                    <td>Number con el nº de elementos</td>
                    <td>- La categoría no puede ser null.<br/>
                        - La categoría ya existe</td>
                </tr>
                <tr class="fila">
                    <td>removeCategory</td>
                    <td>Elimina una categoría</td>
                    <td>Objeto Category</td>
                    <td>Number con el nº de elementos</td>
                    <td>- La categoría no está registrada</td>
                </tr>
                <tr class="fila">
```

```
                    <td>addImage</td>
                    <td>Añade una nueva imagen a una categoría con un autor.</td>
                    <td>- Objeto Image<br/>
                        - Objeto Category<br/>
                        - Objeto Author<br/></td>
                    <td>Number con el nº de elementos</td>
                    <td>- La imagen no puede ser null.</td>
                </tr>
                <tr class="fila">
                    <td>removeImage</td>
                    <td>Elimina una imagen</td>
                    <td>Objeto Image</td>
                    <td>Number con el nº de elementos</td>
                    <td>- La imagen no existe.</td>
                </tr>
                <tr class="fila">
                    <td>getCategoryImages</td>
                    <td>Devuelve todas las imágenes de una determinada categoría</td>
                    <td>Objeto Category</td>
                    <td>[]Images</td>
                    <td>- La categoría no puede ser null.</td>
                </tr>
                <tr class="fila">
                    <td>addAuthor</td>
                    <td>Añade un nuevo autor a la Galería</td>
                    <td>Objeto Author</td>
                    <td>Number con el nº de autores.</td>
                    <td>- El autor no puede ser nulo</td>
                </tr>
                <tr class="fila">
                    <td>removeAuthor</td>
                    <td>Eliminar un autor</td>
                    <td>Objeto Author</td>
                    <td>Number con el nº de elementos</td>
                    <td>- Author no existente.</td>
                </tr>
                <tr class="fila">
                    <td>getAuthorImages</td>
                    <td>Devuelve todas las imágenes añadidas por un autor.</td>
                    <td>Objeto Author</td>
                    <td>[]Images</td>
                    <td>- El autor no puede ser null.</td>
                </tr>
                <tr class="fila">
                    <td>getPortraits</td>
                    <td>Devuelve todas las imágenes de tipo retrato</td>
                    <td>-</td>
                    <td>[]Portrait</td>
                    <td>-</td>
                </tr>
                <tr class="fila">
                    <td>getLandscapes</td>
                    <td>Devuelve todas las imágenes de tipo apaisado</td>
                    <td>-</td>
                    <td>[]Landscape</td>
                    <td>-</td>
                </tr>
            </table>
            <h3>2. Objeto Category</h3>
            <h4>Con este objeto podemos crear la estructura de categorías de imágenes de la
 galería. Sus propiedades serán:</h4>
            <h4>- Title: Con el título de la categoría.<br/>
                - Description: Con la descripción de la misma.</h4>
            <h4>Sus métodos serán los habituales getter y setter de estas propiedades, aunq
ue el título no podrá ser vacío.</h4>
```

```
            <h3>3. Objeto Image</h3>
            <h4>Objeto que permitirá añadir imágenes a la galería. Tendrá las siguientes pr
opiedades:</h4>
            <h4>- Title: Título de la imagen. Obligatoria.<br/>
                - Description: Descripción de la imagen.<br/>
                - URL: Dirección donde está ubicada la imagen. Obligatoria.<br/>
                - Coords: Coordenadas donde se tomó la imagen.</h4>
            <h4>Como métodos tendrá los getter y setter habituales.</h4>
            <h3>3.1. Objeto Landscape</h3>
            <h4>Hereda de Image para especializar una imagen apaisada.<br/>
                No tiene propiedades específicas.</h4>
            <h3>3.2. Objeto Portrait</h3>
            <h4>Hereda de Image para especilizar una imagen en forma de retrato.<br/>
                No tiene propiedades específicas.</h4>
            <h3>4. Objeto Coords</h3>
            <h4>Coordenadas para adjuntar a una imagen. Sus propiedades son:</h4>
            <h4>- Latitude: Latitud.<br/>
                - Longitude: Longitud.</h4>
            <h4>Los métodos serán getter y setter, siendo ambas propiedades obligatorias.</
h4>
            <h3>5. Objeto Author</h3>
            <h4>Información del autor de la imagen. Sus propiedades son:</h4>
            <h4>- Nickname: Nombre del autor. Obligatorio.<br/>
                - Email: Correo electrónico. Obligatorio.<br/>
                - Avatar: imagen asociada al usuario.</h4>
            <h4>Como métodos tendrá los getter y setter habituales teniendo en cuenta las p
ropiedades que son obligatorias.</h4>
        </div>
        <br/>
        <!-- En este div resuelvo el ejercicio. -->
        <div class="ejercicio">
            <p>Para comprobar el correcto funcionamiento de cada una de las funciones imple
mentadas, he desarollado una función de testeo en un archivo .js independiente en el que se
 deja un registro de cada una de las operación a través de la consola. <br/>
            Para acceder a la consola:<br/>
            - En Google Chrome, pulsar Ctrl + Shift + j.<br/>
            - En Mozilla Firefox, pulsar Ctrl + Shift + k.<br/>
            - En Opera, pulsar Ctrl + Shift + j.<br/>
            - En Microsoft Edge, pulsar Ctrl + Shift + j.
            <br/>
            </p>
        </div>
        <!-- Indico la ruta del código javascript. -->
        <script src="Excepciones.js"></script>
        <script src="Gallery.js"></script>
        <script src="Category.js"></script>
        <script src="Image.js"></script>
        <script src="Author.js"></script>
        <script src="Landscape.js"></script>
        <script src="Portrait.js"></script>
        <script src="Coords.js"></script>
        <script src="Testeo.js"></script>
    </body>
</html>
```

Mayorga_Usero_JuanJose_DWEC04_hojadeestilo.css

Una hoja de estilos CSS para dar formato a la página web.

```
/* Cambio la fuente por defecto y aplico un color al fondo. */
body
{
```

```
    font-family: Helvetica, sans-serif;
    background-color: cadetblue;
}

/* Estilo de la cabecera: Centrada, con borde redondeado. */
.cabecera
{
    width: 80%;
    margin: auto;
    border-style:solid;
    border-width:5px;
    border-color:black;
    background-color:black;
    color:white;
    border-radius:20px;
    text-align: center;
}

/* Estilo del eneunciado: Centrada, con borde redondeado.*/
.enunciado
{
    width: 80%;
    margin: auto;
    padding: 10px;
    border-style:solid;
    border-width: 5px;
    border-color: navy;
    background-color: powderblue;
    border-radius:20px;
}

/* Estilo del ejercicio: Centrada, con borde redondeado. */
.ejercicio
{
    width: 80%;
    margin: auto;
    padding: 10px;
    border-style:solid;
    border-width: 5px;
    border-color: navy;
    background-color: powderblue;
    border-radius:20px;
}

.filacabeza
{
    background-color:#000066;
    color:white;
```

```
}

.fila:nth-child(odd)
{
    background-color: #b3b3ff;
}
.fila:nth-child(even)
{
    background-color: #e6e6ff;
}
```

Excepciones.js

Con este archivo manejaré las excepciones que se indican en el enunciado.

```
/*
Juan José Mayorga Usero
CFGS Desarrollo de Aplicaciones Web
I.E.S. Maestre de Calatrava - Ciudad Real
Desarrollo Web en Entorno Cliente
UT04 Práctica 2
*/

function BaseException(message = "Default Message", fileName, lineNumber)
{
    let instance = new Error(message, fileName, lineNumber);
    instance.name = "MyError";
    Object.setPrototypeOf(instance, Object.getPrototypeOf(this));
    if(Error.captureStackTrace)
    {
        Error.captureStackTrace(instance, BaseException);
    }
    return instance;
}
BaseException.prototype = Object.create(Error.prototype,
{
    constructor:
    {
        value: BaseException,
        enumerable: false,
        writable: true,
        configurable: true
    }
});

function ConstructorInvalido()
{
    let instance = BaseException.call(this, "El constructor del objeto no se ha usado adecu
adamente.");
    instance.name = "ConstructorInvalido";
    return instance;
}
ConstructorInvalido.prototype = Object.create(BaseException.prototype);
ConstructorInvalido.prototype.constructor = ConstructorInvalido;

function TituloVacio()
{
    let instance = BaseException.call(this, "El título no puede ser vacío.");
    instance.name = "TituloVacio";
    return instance;
```

```
}
TituloVacio.prototype = Object.create(BaseException.prototype);
TituloVacio.prototype.constructor = TituloVacio;

function DescripcionVacia()
{
    let instance = BaseException.call(this, "La descripción no puede ser vacía.");
    instance.name = "DescripcionVacia";
    return instance;
}
DescripcionVacia.prototype = Object.create(BaseException.prototype);
DescripcionVacia.prototype.constructor = DescripcionVacia;

function URLVacia()
{
    let instance = BaseException.call(this, "La URL no puede ser vacía.");
    instance.name = "URLVacia";
    return instance;
}
URLVacia.prototype = Object.create(BaseException.prototype);
URLVacia.prototype.constructor = URLVacia;

function NombreVacio()
{
    let instance = BaseException.call(this, "El nombre no puede ser vacío.");
    instance.name = "NombreVacio";
    return instance;
}
NombreVacio.prototype = Object.create(BaseException.prototype);
NombreVacio.prototype.constructor = NombreVacio;

function EmailVacio()
{
    let instance = BaseException.call(this, "El e-mail no puede ser vacío.");
    instance.name = "EmailVacio";
    return instance;
}
EmailVacio.prototype = Object.create(BaseException.prototype);
EmailVacio.prototype.constructor = EmailVacio;

function CategoriaInvalida()
{
    let instance = BaseException.call(this, "La categoría debe ser un objeto de la clase Ca
tegory.");
    instance.name = "CategoriaInvalida";
    return instance;
}
CategoriaInvalida.prototype = Object.create(BaseException.prototype);
CategoriaInvalida.prototype.constructor = CategoriaInvalida;

function CategoriaNull()
{
    let instance = BaseException.call(this, "La categoría no puede ser null.");
    instance.name = "CategoriaNull";
    return instance;
}
CategoriaNull.prototype = Object.create(BaseException.prototype);
CategoriaNull.prototype.constructor = CategoriaNull;

function CategoriaExiste()
{
    let instance = BaseException.call(this, "La categoría ya existe.");
    instance.name = "CategoriaExiste";
    return instance;
}
```

```
CategoriaExiste.prototype = Object.create(BaseException.prototype);
CategoriaExiste.prototype.constructor = CategoriaExiste;

function CategoriaNoRegistrada()
{
    let instance = BaseException.call(this, "La categoría no está registrada.");
    instance.name = "CategoriaNoRegistrada";
    return instance;
}
CategoriaNoRegistrada.prototype = Object.create(BaseException.prototype);
CategoriaNoRegistrada.prototype.constructor = CategoriaNoRegistrada;

function ImagenInvalida()
{
    let instance = BaseException.call(this, "La imagen debe ser un objeto de la clase Image
.");
    instance.name = "ImagenInvalida";
    return instance;
}
ImagenInvalida.prototype = Object.create(BaseException.prototype);
ImagenInvalida.prototype.constructor = ImagenInvalida;

function ImagenNull()
{
    let instance = BaseException.call(this, "La imagen no puede ser null.");
    instance.name = "ImagenNull";
    return instance;
}
ImagenNull.prototype = Object.create(BaseException.prototype);
ImagenNull.prototype.constructor = ImagenNull;

function ImagenExiste()
{
    let instance = BaseException.call(this, "La imagen ya existe.");
    instance.name = "ImagenExiste";
    return instance;
}
ImagenExiste.prototype = Object.create(BaseException.prototype);
ImagenExiste.prototype.constructor = ImagenExiste;

function ImagenNoExiste()
{
    let instance = BaseException.call(this, "La imagen no existe.");
    instance.name = "ImagenNoExiste";
    return instance;
}
ImagenNoExiste.prototype = Object.create(BaseException.prototype);
ImagenNoExiste.prototype.constructor = ImagenNoExiste;

function AutorInvalido()
{
    let instance = BaseException.call(this, "El autor debe ser un objeto de la clase Author
.");
    instance.name = "AutorInvalido";
    return instance;
}
AutorInvalido.prototype = Object.create(BaseException.prototype);
AutorInvalido.prototype.constructor = AutorInvalido;

function AutorNull()
{
    let instance = BaseException.call(this, "El autor no puede ser nulo.");
    instance.name = "AutorNull";
    return instance;
}
```

```
AutorNull.prototype = Object.create(BaseException.prototype);
AutorNull.prototype.constructor = AutorNull;

function AutorNoExiste()
{
    let instance = BaseException.call(this, "El autor no existe.");
    instance.name = "AutorNoExiste";
    return instance;
}
AutorNoExiste.prototype = Object.create(BaseException.prototype);
AutorNoExiste.prototype.constructor = AutorNoExiste;

function AutorExiste()
{
    let instance = BaseException.call(this, "El autor ya existe en la galería.");
    instance.name = "AutorExiste";
    return instance;
}
AutorExiste.prototype = Object.create(BaseException.prototype);
AutorExiste.prototype.constructor = AutorExiste;

function CoordsInvalidas()
{
    let instance = BaseException.call(this, "Las coordenadas introducidas no es un objeto C
oords válido.");
    instance.name = "CoordsInvalidas";
    return instance;
}
CoordsInvalidas.prototype = Object.create(BaseException.prototype);
CoordsInvalidas.prototype.constructor = CoordsInvalidas;
```

Testeo.js

Finalmente, cuando ya tengo implementados todos los archivos necesarios, creo una función de testeo, en la que voy a crear una galería de ejemplo y voy a probar las diversas funcionalidades implementadas.

```
/*
Juan José Mayorga Usero
CFGS Desarrollo de Aplicaciones Web
I.E.S. Maestre de Calatrava - Ciudad Real
Desarrollo Web en Entorno Cliente
UT04 Práctica 2
*/

function testeo()
{
    // En primer lugar, voy a crear varios objetos de los tipos que se piden en el enunciad
o.
    console.log("En primer lugar, voy a crear varios objetos de los tipos que se piden en e
l enunciado.");
    // Objetos Image: Creo 6 objetos Image. Se pueden mostrar por consola directamente.
    console.log("Objetos Image: Creo 6 objetos Image. Se pueden mostrar por consola directa
mente.");
    let imagen1 = new Image("Remontada al PSG", "Messi celebrando la victoria frente al PSG
", "https://e00-
marca.uecdn.es/assets/multimedia/imagenes/2017/04/03/14912490601797.jpg", new Coords(41.39,
2.17));
    let imagen2 = new Image("The last shot", "Michael Jordan anotando frente a Utah Jazz",
"https://e00-
marca.uecdn.es/assets/multimedia/imagenes/2018/06/14/15289587333620.jpg", new Coords(40.76,
-111.89));
```

```
    let imagen3 = new Image("La mano de Dios", "Maradona marca a Inglaterra con la mano", "
https://cdn2.mediotiempo.com/uploads/media/2019/06/22/mano-dios-gol-maradona-
inglaterra.jpg", new Coords(19.30,-99.15));
    let imagen4 = new Image("Vinsanity", "Vince Carter en el concurso de mates de 2000", "h
ttps://www.bardown.com/polopoly_fs/1.1441690!/fileimage/httpImage/image.jpg_gen/derivatives
/default/vc1.jpg", new Coords(37.75,-122.20));
    let imagen5 = new Image("El más rápido", "Usain Bolt bate el récord mundial de 100m lis
os", "https://e00-
marca.uecdn.es/assets/multimedia/imagenes/2019/08/15/15658714229821.jpg", new Coords(52.52,
13.38));
    let imagen6 = new Image("El hijo del viento", "Carl Lewis en Barcelona 92", "https://im
g.olympicchannel.com/images/image/private/t_16-
9_760/v1538355600/primary/adc9oztu46zmomtj7sio", new Coords(41.39,2.17));
    console.log(imagen1);
    console.log(imagen2);
    console.log(imagen3);
    console.log(imagen4);
    console.log(imagen5);
    console.log(imagen6);
    // Objetos Landscape: Creo 3 objetos Landscape. Se pueden mostrar por consola directame
nte.
    console.log("Objetos Landscape: Creo 3 objetos Landscape. Se pueden mostrar por consola
 directamente.");
    let landscape1 = new Landscape("El nido de pájaro", "Estadio Olímpico de Pekín", "https
://d1xymaf218jlo1.cloudfront.net/2015/12/pekin-estadio.jpg", new Coords(39.65,116.24));
    let landscape2 = new Landscape("Tower Bridge", "El puente de Londres con los aros olímp
icos", "https://i.pinimg.com/originals/ee/e6/b6/eee6b669afe00f2317db22eecf386d5e.jpg", new
Coords(51.51,-0.09));
    let landscape3 = new Landscape("Wimbledon", "Pistas de juego del torneo de Wimbledon",
"https://e00-
marca.uecdn.es/assets/multimedia/imagenes/2020/03/27/15853083295304.jpg", new Coords(51.51,
-0.09));
    console.log(landscape1);
    console.log(landscape2);
    console.log(landscape3);
    // Objetos Portrait: Creo 2 objetos Portrait. Se pueden mostrar por consola directament
e.
    console.log("Objetos Portrait: Creo 2 objetos Portrait. Se pueden mostrar por consola d
irectamente.");
    let portrait1 = new Portrait("La saeta rubia", "Retrato de Alfredo Di Stéfano", "https:
//s3.amazonaws.com/arc-wordpress-client-uploads/infobae-wp/wp-
content/uploads/2017/11/25171611/GettyImages-3429336.jpg", new Coords(41.39,2.17));
    let portrait2 = new Portrait("Black mamba", "Retrato de Kobe Bryant con el trofeo de la
 NBA", "https://cdn.shopify.com/s/files/1/1050/9876/products/kobe6_1200x1200.jpg?v=15801592
20", new Coords(34.04,-118.27));
    console.log(portrait1);
    console.log(portrait2);
    // Objetos Category: Creo 4 objetos Category. Se pueden mostrar por consola directament
e.
    console.log("Objetos Category: Creo 4 objetos Category. Se pueden mostrar por consola d
irectamente.");
    let categoria1 = new Category("Fútbol");
    let categoria2 = new Category("Baloncesto");
    let categoria3 = new Category("Atletismo");
    let categoria4 = new Category("Tenis");
    console.log(categoria1);
    console.log(categoria2);
    console.log(categoria3);
    console.log(categoria4);
    // Objetos Author: Creo 4 objetos Author. Se pueden mostrar por consola directamente.
    console.log("Objetos Author: Creo 4 objetos Author. Se pueden mostrar por consola direc
tamente.");
    let autor1 = new Author("Santiago Garcés", "sgarces@fcbarcelona.com");
    let autor2 = new Author("Fernando Medina", "fmedina@orlandomagic.com");
    let autor3 = new Author("Eduardo Longoni", "elongoni@clarin.com");
```

```
    let autor4 = new Author("Jesse Garrabrant", "jgarrabrant@olympics.com");
    console.log(autor1);
    console.log(autor2);
    console.log(autor3);
    console.log(autor4);
    /*
    Creo un objeto Gallery, con el que trabajaré el resto de la práctica.
    Desarrollando esta función de testeo, he observado que al invocar el objeto galeria1 co
n el comando console.log()
    visualizamos el estado final del objeto, no el estado el objeto en el momento que se in
voca.
    Por ese motivo he creado una función imprimir() para los objetos Gallery, que muestra l
os arrays de objetos Image, Category y Author.
    */
    console.log("Creo un objeto Gallery, con el que trabajaré el resto de la práctica.");
    let galeria1 = new Gallery("Galería de ejemplo");
    // Pruebo la función addCategory, añadiendo el objeto categoria1.
    console.log("Pruebo la función addCategory, añadiendo el objeto categoria1.");
    galeria1.addCategory(categoria1);
    // Si intento añadir la misma categoría dos veces, salta una excepción.
    console.log("Si intento añadir la misma categoría dos veces, salta una excepción.");
    galeria1.addCategory(categoria1);
    galeria1.addCategory(categoria2);
    galeria1.addCategory(categoria3);
    // Muestro galeria1 con 3 categorías añadidas.
    console.log("Muestro galeria1 con 3 categorías añadidas.");
    galeria1.imprimir();
    // Pruebo la función removeCategory. Quito categoria3 y muestro la galería.
    console.log("Pruebo la función removeCategory. Quito categoria3 y muestro la galería.")
;
    galeria1.removeCategory(categoria3);
    galeria1.imprimir();
    // Intento quitar una categoría que no está en la galería. Salta una excepción.
    console.log("Intento quitar una categoría que no está en la galería. Salta una excepció
n.");
    galeria1.removeCategory(categoria4);
    // Finalmente añado categoria3 y categoria4.
    console.log("Finalmente añado categoria3 y categoria4.");
    galeria1.addCategory(categoria3);
    galeria1.addCategory(categoria4);
    galeria1.imprimir();
    // Repito el proceso con los autores. Añado el objeto autor1.
    console.log("Repito el proceso con los autores. Añado el objeto autor1.");
    galeria1.addAuthor(autor1);
    // Si intento añadir nuevamente autor1, salta una excepción.
    console.log("Si intento añadir nuevamente autor1, salta una excepción.");
    galeria1.addAuthor(autor1);
    galeria1.addAuthor(autor2);
    galeria1.addAuthor(autor3);
    // Muestro galeria1 con 3 autores añadidos.
    console.log("Muestro galeria1 con 3 autores añadidos.");
    galeria1.imprimir();
    // Pruebo la función removeAuthor. Quito autor3 y muestro la galería.
    console.log("Pruebo la función removeAuthor. Quito autor3 y muestro la galería.");
    galeria1.removeAuthor(autor3);
    galeria1.imprimir();
    // Intento quitar un autor que no está en la galería. Salta una excepción.
    console.log("Intento quitar un autor que no está en la galería. Salta una excepción.");
    galeria1.removeAuthor(autor4);
    // Finalmente añado autor3 y autor4.
    console.log("Finalmente añado autor3 y autor4.");
    galeria1.addAuthor(autor3);
    galeria1.addAuthor(autor4);
    galeria1.imprimir();
```

```
    // Pruebo la función addImage con la imagen1, que pertenece a categoria1 y es del autor
1.
    console.log("Pruebo la función addImage con la imagen1, que pertenece a categoria1 y es
 del autor1.");
    galeria1.addImage(imagen1,categoria1,autor1);
    galeria1.imprimir();
    // Si intento añadir nuevamente imagen1, salta una excepción.
    console.log("Si intento añadir nuevamente imagen1, salta una excepción.");
    galeria1.addImage(imagen1,categoria1,autor1);
    // Añado algunas imágenes más y las muestro.
    console.log("Añado algunas imágenes más y las muestro.");
    galeria1.addImage(imagen2,categoria2,autor2);
    galeria1.addImage(imagen3,categoria1,autor3);
    galeria1.addImage(imagen4,categoria2,autor4);
    galeria1.addImage(imagen5,categoria3,autor1);
    galeria1.imprimir();
    // Pruebo la función removeImage. Quito imagen5 y muestro la galería.
    console.log("Pruebo la función removeImage. Quito imagen5 y muestro la galería.");
    galeria1.removeImage(imagen5);
    galeria1.imprimir();
    // Intento quitar un autor que no está en la galería. Salta una excepción.
    console.log("Intento quitar una imagen que no está en la galería. Salta una excepción."
);
    galeria1.removeImage(imagen6);
    // Añado de nuevo imagen5. También añado imagen6 y los objetos Portrait y Landscape que
 he creado.
    console.log("Añado de nuevo imagen5. También añado imagen6 y los objetos Portrait y Lan
dscape que he creado.");
    galeria1.addImage(imagen5,categoria3,autor1);
    galeria1.addImage(imagen6,categoria3,autor3);
    galeria1.addImage(landscape1,categoria3,autor4);
    galeria1.addImage(landscape2,categoria3,autor4);
    galeria1.addImage(landscape3,categoria4,autor1);
    galeria1.addImage(portrait1,categoria1,autor3);
    galeria1.addImage(portrait2,categoria2,autor2);
    galeria1.imprimir();
    // Pruebo la función getCategoryImages. Esta función devuelve un array con las imágenes
 que corresponden a la categoría que se le indica como argumento de entrada.
    console.log("Pruebo la función getCategoryImages. Esta función devuelve un array con la
s imágenes que corresponden a la categoría que se le indica como argumento de entrada.");
    // Comienzo hallando las imágenes de categoria1: 'Fútbol'.
    console.log("Comienzo hallando las imágenes de categoria1: 'Fútbol'.");
    let imagenesFutbol = galeria1.getCategoryImages(categoria1);
    // Puedo ver el array con el comando console.log.
    console.log("Puedo ver el array con el comando console.log.");
    console.log(imagenesFutbol);
    // Hago lo mismo con las otras categorías.
    console.log("Hago lo mismo con las otras categorías.");
    let imagenesBaloncesto = galeria1.getCategoryImages(categoria2);
    console.log(imagenesBaloncesto);
    let imagenesAtletismo = galeria1.getCategoryImages(categoria3);
    console.log(imagenesAtletismo);
    let imagenesTenis = galeria1.getCategoryImages(categoria4);
    console.log(imagenesTenis);
    // Pruebo la función getCategoryImages. Esta función devuelve un array con las imágenes
 que corresponden a la categoría que se le indica como argumento de entrada.
    console.log("Pruebo la función getAuthorImages. Esta función devuelve un array con las
imágenes que corresponden al autor que se le indica como argumento de entrada.");
    // Comienzo hallando las imágenes de autor1: 'Fútbol'.
    console.log("Comienzo hallando las imágenes de categoria1: 'Santiago Garcés'.");
    let imagenesGarces = galeria1.getAuthorImages(autor1);
    // Puedo ver el array con el comando console.log.
    console.log("Puedo ver el array con el comando console.log.");
    console.log(imagenesGarces);
    // Hago lo mismo con los otros autores.
```

```
    console.log("Hago lo mismo con los otros autores.");
    let imagenesMedina = galeria1.getAuthorImages(autor2);
    console.log(imagenesMedina);
    let imagenesLongoni = galeria1.getAuthorImages(autor3);
    console.log(imagenesLongoni);
    let imagenesGarrabrant = galeria1.getAuthorImages(autor4);
    console.log(imagenesGarrabrant);
    // Pruebo la función getPortraits. Esta función devuelve un array con los objetos Portr
ait (o mejor dicho, las imágenes cuya imagen es de tipo Portrait).
    console.log("Pruebo la función getPortraits. Esta función devuelve un array con los obj
etos Portrait (o mejor dicho, las imágenes cuya imagen es de tipo Portrait).");
    let portraits1 = galeria1.getPortraits();
    console.log(portraits1);
    // Pruebo la función getLandscapes. Esta función devuelve un array con los objetos Land
scape (o mejor dicho, las imágenes cuya imagen es de tipo Landscape).
    console.log("Pruebo la función getLandscapes. Esta función devuelve un array con los ob
jetos Landscape (o mejor dicho, las imágenes cuya imagen es de tipo Landscape).");
    let landscapes1 = galeria1.getLandscapes();
    console.log(landscapes1);
    // Finalmente, puedo mostrar el objeto galeria1 con console.log. Este comando nos da el
 estado final del objeto.
    console.log("Finalmente, puedo mostrar el objeto galeria1 con console.log. Este comando
 nos da el estado final del objeto.");
    console.log(galeria1);
}

window.onload = testeo();
```